

Docket No. RSW920010011US1

**METHOD AND SYSTEM FOR SPECIFYING A CACHE POLICY FOR CACHING
WEB PAGES WHICH INCLUDE DYNAMIC CONTENT**

1. Field of the Invention:

5 The present invention relates generally to data
processing systems, and more particularly to caching data
in a data processing system. Still more particularly,
the present invention relates to a data processing
system, method, and product for caching WEB pages which
10 include dynamic content.

2. Background of the Invention:

 The Internet, also referred to as an "internetwork",
is a set of computer networks, possibly dissimilar, joined
15 together by means of gateways that handle data transfer
and the conversion of messages from the sending network to
the protocols used by the receiving network (with packets
if necessary). When capitalized, the term "Internet"
refers to the collection of networks and gateways that use
20 the TCP/IP suite of protocols.

 The Internet has become a cultural fixture as a
source of both information and entertainment. Many
businesses are creating Internet sites as an integral part
of their marketing efforts, informing consumers of the
25 products or services offered by the business or providing
other information seeking to engender brand loyalty. Many
federal, state, and local government agencies are also
employing Internet sites for informational purposes,
particularly agencies which must interact with virtually

Docket No. RSW920010011US1

all segments of society such as the Internal Revenue Service and secretaries of state. Providing informational guides and/or searchable databases of online public records may reduce operating costs. Further, the Internet
5 is becoming increasingly popular as a medium for commercial transactions.

Currently, the most commonly employed method of transferring data over the Internet is to employ the World Wide Web environment, also called simply "the Web". Other
10 Internet resources exist for transferring information, such as File Transfer Protocol (FTP) and Gopher, but have not achieved the popularity of the Web. In the Web environment, servers and clients effect data transaction using the Hypertext Transfer Protocol (HTTP), a known
15 protocol for handling the transfer of various data files (e.g., text, still graphic images, audio, motion video, etc.). The information in various data files is formatted for presentation to a user by a standard page description language, the Hypertext Markup Language
20 (HTML). In addition to basic presentation formatting, HTML allows developers to specify "links" to other Web resources identified by a Uniform Resource Locator (URL). A URL is a special syntax identifier defining a communications path to specific information. Each logical
25 block of information accessible to a client, called a "page" or a "Web page", is identified by a URL. The URL provides a universal, consistent method for finding and accessing this information, not necessarily for the user,

Docket No. RSW920010011US1

but mostly for the user's Web "browser". A URL includes a Uniform Resource Identifier (URI). The URI is the portion of the URL which more specifically identifies a particular page to be displayed.

5 A browser is a program capable of submitting a request for information identified by a URL at the client machine. Retrieval of information on the Web is generally accomplished with an HTML-compatible browser.

Web content is often dynamic. In the modern
10 Internet, personalization of content to specific users and groups necessitates dynamic content, as does changing content due to user actions (e.g. shopping carts change, though your request for that cart does not). Even static pages are occasionally updated. Web servers provide
15 static content and dynamic content to various users. Static content contain data from files stored at a server. Dynamic content is constructed by programs, including such technologies as servlets, ASPs, and CGI, executing at the time a request is made. The presence of dynamic content
20 often slows down Web sites considerably. High-performance Web servers can typically deliver several hundred static pages per second. By contrast, the rate at which dynamic pages are delivered is often one or two order of magnitudes slower.

25 Dynamic content is often present at a Web site in an effort to provide customized pages and updated information to various users that may visit the site. The use of this

Docket No. RSW920010011US1

type of Web page, however, may cause a Web site to slow down in performance.

In the generic web application environment, dynamic content is generated (e.g. by executing a servlet) for every request. A dynamic web cache allows a dynamically generated page to be cached and later served in response to future requests without regenerating its output (without executing that servlet again). The first time a request is made for dynamic content, the application executes the appropriate servlets necessary to display the page. The output of these servlets is typically HTML code which is then presented to the user. Other types output include XML and images such as GIFs and JPGs. When a user requests a page for the first time, the servlets execute and the code is stored as a cache entry. Each subsequent time the user requests this page, this cache entry is retrieved and presented to the user. When the page is to be refreshed, the application executes all of the servlets again to create a new cache entry.

The method described above for caching dynamic content can be applied to entire pages, requested externally by users. This method is inflexible, and often inefficient, as whole pages are generally constructed from several dynamic fragments. Frequently, the content of only parts of a page may change. In these cases, valuable computing resources are wasted by regenerating those parts of the page which were not changed.

Docket No. RSW920010011US1

Some applications include a caching capability within the application itself. Each application may choose to implement caching in a unique manner. The form of the servlets will vary from one application to the next. In these systems, each servlet must know how to generate its own cache entry. Therefore, in order to change the way the dynamic content is cached, each servlet, in each application, must be changed. Further, with this approach, each application must provide for its own caching, which cannot be applied to other applications. Existing applications which do not currently provide for caching must be updated in order to permit caching.

In these systems, some servlets are cacheable, and some are not. When a servlet is cacheable, the servlet includes the information necessary to generate its cache entry.

Therefore, a need exists for a data processing system and method for specifying a caching policy for caching dynamic content including portions of pages and supporting both internal and external requests, where caching is executed separately from applications.

Docket No. RSW920010011US1

SUMMARY OF THE INVENTION

A data processing system and method are described
5 for specifying a cache policy for caching pages which
include dynamic content. A user is permitted to request
one of the pages to be displayed. The page includes
multiple fragments. An application is executed which
includes multiple servlets, each of which generates a
10 dynamic fragment. The servlets are unchanged by the
caching policy. Each one of the servlets is executed to
present a different one of the fragments. Caching of the
page fragments is processed separately from the execution
of the application and its servlets.

15 The above as well as additional object, features,
and advantages of the present invention will become
apparent in the following detailed written description.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 depicts a pictorial representation of a distributed data processing system in which the present invention may be implemented;

Figure 2 is a block diagram of a server system depicted in accordance with a preferred embodiment of the present invention;

Figure 3 is a block diagram depicting a data processing system that may be implemented as a server in accordance with a preferred embodiment of the present invention;

Figure 4 is a block diagram illustrating a data processing system in which the present invention may be implemented;

Figure 5 is a diagram illustrating examples of different update rates and caching for pages depicted in accordance with a preferred embodiment of the present invention;

Docket No. RSW920010011US1

Figure 6 is a diagram illustrating page fragments depicted in accordance with a preferred embodiment of the present invention;

Figure 7 is a high level flow chart which depicts
5 the creation of a servlet and a servlet element which uniquely identifies a cacheable servlet in accordance with the present invention;

Figure 8 is a high level flow chart which
illustrates processing an external request for a servlet
10 in accordance with the present invention; and

Figure 9 is a high level flow chart which
illustrates processing an internal request for a servlet
in accordance with the present invention.

Docket No. RSW920010011US1

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A preferred embodiment of the present invention and its advantages are better understood by referring to the figures, like numerals being used for like and
5 corresponding parts of the accompanying figures.

The invention is preferably realized using a well-known computing platform, such as an IBM RS/6000 workstation running the IBM AIX operating system. However, it may be realized in other popular computer
10 system platforms, such as an IBM personal computer running the Microsoft Windows operating system or a Sun Microsystems workstation running operating systems such as UNIX or LINUX, without departing from the spirit and scope of the invention.

15 The present invention is a system, method, and product for specifying a cache policy for caching pages which include dynamic content. A user is permitted to request a page to be displayed. The page includes multiple fragments. An application is executed which
20 includes multiple servlets. Each one of the servlets is executed to present a different one of the fragments. Dynamic content is constructed by programs, including such technologies as servlets, ASPs, and CGI, executing at the time a request is made. For the purposes of the following
25 description, it is assumed that servlets are the technology being used to implement dynamic content. However, those skilled in the art will recognize that the present invention may be utilized when any of the

Docket No. RSW920010011US1

different technologies are used to implement dynamic content.

Each servlet was written without regard to caching. The servlets are stored on an application server disk.

5 For each servlet, caching options are specified inside a separate XML file dedicated to the task of storing cache policies. A servlet element is created for each servlet, which uniquely identifies that servlet. The servlet element includes an indication of the caching options for
10 the servlet.

During initialization of the Web Application, the present invention receives a cache policy file. This file contains a listing of servlet elements. Each
15 servlet element specifies a set of caching options and specifies a servlet to which those options apply. From that file it will build a list in memory of servlets that are to be cached, and associate the specified options with those servlets.

When a user requests a document to be displayed, the
20 servlets that generate that document are initialized, if initialization has not already occurred. Each servlet identified by a servlet element included in the request is located and loaded from disk to the application server's memory. For each servlet, if it is to be
25 cached, the cache options specified for it are associated with the servlet in memory.

A cache identifier is then generated for each servlet. The cache options for a servlet describe how to

Docket No. RSW920010011US1

use the information included in the document request to build a cache identifier.

5 The present invention then searches to attempt to locate a cache entry which is identified by the newly generated cache identifier. The first time a user requests an output which is generated by a particular servlet, no cache entry will be found. In this case, a cache entry must be generated. To generate a cache entry for a particular servlet, the servlet is executed and
10 generates output, typically HTML. This output is saved utilizing the cache identifier as the cache entry's identifier. This output is then returned and presented to the user.

15 Subsequent times the user requests the output which is generated by the particular servlet, the present invention will be able to locate that cache entry having the particular cache identifier. In these cases, the contents of the cache entry are then retrieved and the output is returned and presented to the user.

20 A user may only request an entire page, but the execution of that page may involve requesting additional dynamic fragments. Since each fragment may have its own request, each fragment will have its own cache entry. When a requested fragment does not have an associated
25 cache entry, a cache entry is created by executing only the servlet associated with the fragment. In this manner, the entire page need not be cached in order to update only a portion of the page.

Docket No. RSW920010011US1

The present invention provides for specifying and executing caching policies separately from the execution of an application. In this manner, the caching policy may be applied to existing applications. The servlets of
5 the existing applications need not be modified in order to execute the present invention.

With reference now to the figures, **Figure 1** depicts a pictorial representation of a distributed data processing system in which the present invention may be implemented.
10 Distributed data processing system 100 is a network of computers in which the present invention may be implemented. Distributed data processing system 100 contains a network 102, which is the medium used to provide communications links between various devices and
15 computers connected together within distributed data processing system 100. Network 102 may include permanent connections, such as wire or fiber optic cables, or temporary connections made through telephone connections.

In the depicted example, a server 104 is connected to
20 network 102 along with storage unit 106. In addition, clients 108, 110, and 112 also are connected to a network 102. These clients 108, 110, and 112 may be, for example, personal computers or network computers. For purposes of this application, a network computer is any computer,
25 coupled to a network, which receives a program or other application from another computer coupled to the network. In the depicted example, server 104 provides data, such as boot files, operating system images, and applications to

Docket No. RSW920010011US1

clients 108-112. Clients 108, 110, and 112 are clients to server 104. Distributed data processing system 100 includes a server system 114, which also may provide data to clients 108-112. Server system 114 may take various forms. For example, server system 114 may consist of two or more servers that have been logically associated with each other or may actually be interconnected as a cluster. Distributed data processing system 100 may include additional servers, clients, and other devices not shown.

In the depicted example, distributed data processing system 100 is the Internet with network 102 representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, distributed data processing system 100 also may be implemented as a number of different types of networks, such as, for example, an intranet, a local area network (LAN), or a wide area network (WAN). **Figure 1** is intended as an example, and not as an architectural limitation for the present invention.

With reference now to **Figure 2**, a block diagram of a server system is depicted in accordance with a preferred embodiment of the present invention. Server system 200 is an example of a server system that may be implemented as server system 114 in **Figure 1**. Server system 200

Docket No. RSW920010011US1

illustrates a mechanism in which a Web application achieves throughput scale up and high availability. In this example, server system 200 contains Web server nodes 202 and 204, which execute Web application servers 206 and 208 respectively. The use of the term "server" may refer to either the physical data processing system or to an application providing receiving and processing requests from a client. Similarly, the client may be a physical data processing system or an application. For example, a client application would be a Web browser.

Web server nodes 202 and 204 typically coordinate via shared data (e.g., a distributed file system or database system). These data mechanisms can themselves scale up via their own form of parallelism. In the depicted example, this sharing of data is accomplished through shared parallel database 212, which contains an interconnect 214 providing a connection between storage devices 216-222. These storage devices form a cluster and are shared by both Web server node 202 and Web server node 204. These storage devices contain relational databases from which content is pulled by Web server nodes 202 and 204 to dynamically create pages.

The depicted configuration for server system 200 is intended as an example and is not meant to imply architectural limitations with respect to the present invention.

Referring to **Figure 3**, a block diagram depicts a data processing system that may be implemented as a server,

Docket No. RSW920010011US1

such as server 104 in Figure 1, Web server node 202, or Web server node 204, in accordance with a preferred embodiment of the present invention. Data processing system 300 may be a symmetric multiprocessor (SMP) system including a plurality of processors 302 and 304 connected to system bus 306. Alternatively, a single processor system may be employed. Also connected to system bus 306 is memory controller/cache 308, which provides an interface to local memory 309. I/O bus bridge 310 is connected to system bus 306 and provides an interface to I/O bus 312. Memory controller/cache 308 and I/O bus bridge 310 may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge 314 connected to I/O bus 312 provides an interface to PCI local bus 316. A number of modems may be connected to PCI bus 316. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers 108-112 in Figure 1 may be provided through modem 318 and network adapter 220 connected to PCI local bus 316 through add-in boards.

Additional PCI bus bridges 322 and 324 provide interfaces for additional PCI buses 326 and 328, from which additional modems or network adapters may be supported. In this manner, server 300 allows connections to multiple network computers. A memory-mapped graphics adapter 330 and hard disk 332 may also be connected to I/O bus 312 as depicted, either directly or indirectly.

Docket No. RSW920010011US1

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 3** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in
5 place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

The data processing system depicted in **Figure 3** may be, for example, an IBM RISC/System 6000 system, a product
10 of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system.

With reference now to **Figure 4**, a block diagram illustrates a data processing system in which the present
15 invention may be implemented. Data processing system 400 is an example of a client computer. Data processing system 400 employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as
20 Micro Channel and ISA may be used. Processor 402 and main memory 404 are connected to PCI local bus 406 through PCI bridge 408. PCI bridge 408 also may include an integrated memory controller and cache memory for processor 402. Additional connections to PCI local bus 406 may be made
25 through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter 410, SCSI host bus adapter 412, and expansion bus interface 414 are connected to PCI local bus 406 by direct

Docket No. RSW920010011US1

component connection. In contrast, audio adapter 416, graphics adapter 418, and audio/video adapter 419 are connected to PCI local bus 406 by add-in boards inserted into expansion slots. Expansion bus interface 414
5 provides a connection for a keyboard and mouse adapter 420, modem 422, and additional memory 424. SCSI host bus adapter 412 provides a connection for hard disk drive 426, tape drive 428, and CD-ROM drive 430. Typical PCI local bus implementations will support three or four PCI
10 expansion slots or add-in connectors.

An operating system runs on processor 402 and is used to coordinate and provide control of various components within data processing system 400 in Figure 4. The operating system may be a commercially available operating
15 system such as OS/2, which is available from International Business Machines Corporation. "OS/2" is a trademark of International Business Machines Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provides calls
20 to the operating system from Java programs or applications executing on data processing system 400. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage
25 devices, such as hard disk drive 426, and may be loaded into main memory 404 for execution by processor 402.

Those of ordinary skill in the art will appreciate that the hardware in Figure 4 may vary depending on the

Docket No. RSW920010011US1

implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in
5 **Figure 4**. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

For example, data processing system 400, if optionally configured as a network computer, may not
10 include SCSI host bus adapter 412, hard disk drive 426, tape drive 428, and CD-ROM 430, as noted by dotted line 432 in **Figure 4** denoting optional inclusion. In that case, the computer, to be properly called a client computer, must include some type of network communication
15 interface, such as LAN adapter 410, modem 422, or the like. As another example, data processing system 400 may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system 400 comprises some
20 type of network communication interface. As a further example, data processing system 400 may be a Personal Digital Assistant (PDA) device which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-
25 generated data.

The depicted example in **Figure 4** and above-described examples are not meant to imply architectural limitations. For example, data processing system 400

Docket No. RSW920010011US1

also may be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system 400 also may be a kiosk or a Web appliance.

5 The present invention provides a method, apparatus, and instructions for caching dynamic content. The mechanism of the present invention is especially useful in caching Web content, such as Web pages. In particular, the mechanism of the present invention may be used to provide caching for Web pages containing data
10 having different rates of change.

Turning to **Figure 5**, a diagram illustrating examples of different update rates and caching for pages is depicted in accordance with a preferred embodiment of the present invention. **Figure 5** shows update rates for
15 various types of pages as well as the type of caching that may be used.

Generally, caching is more beneficial as the access rate increases and as the update rate decreases. All Web content is somewhat dynamic because everything changes
20 eventually, even static pages. For example, if content changes very infrequently, then it is convenient for a human to republish the Web site whenever its content changes. An example of this type of content is a typical home page. It is usually safe to enable browser caching.

25 If content changes so often that it is unreasonable to republish the Web site every time it changes, then a template, such as a Java Server Page (JSP), may be used to dynamically get the content from a file or database,

Docket No. RSW920010011US1

and then render (i.e., format) the content into an HTML page. An ad service is an example of content that may change each time a page is requested. A JSP and a servlet are both mechanisms, which use Java standards for programming dynamic content. A JSP is aimed at Web application designers and servlets are aimed at programmers. A JSP is compiled into a servlet for execution. In this case, static caching in browsers may be disabled, and dynamic caching may or may not be useful.

If the content is constant over a large number of requests, then performance can be significantly improved by using dynamic caching. Examples of content that is fairly consistent are products in e-commerce and white pages as shown in **Figure 5**. With dynamic caching, either time limit or a data ID invalidation mechanism can be used to keep the content in the cache up to date. One way to view this case is that it automates the publishing process so that high update rates can be handled.

If the content changes continuously, such as, for example, a ticker tape, any form of caching is a bad idea because caching overhead is suffered with no benefit. JSPs may be used to generate a page containing this content without any caching.

With reference now to **Figure 6**, a diagram illustrating page fragments is depicted in accordance with a preferred embodiment of the present invention. One mechanism provided by the present invention to provide

Docket No. RSW920010011US1

improved caching of content is to define content in a page as one or more fragments and cache the fragments individually, rather than as a single page.

A fragment is a part or all of a rendered HTML page which can be cached. A fragment can contain 0 or more child fragments, and can be contained by 0 or more parent fragments, forming a directed acyclic graph (DAG). **Figure 6** illustrates a returned page 600, which is a product display page. Page 600 is a "top-level" fragment made up of 5 child fragments. Page 600 includes a product gif URL fragment 602, a product detail fragment 604, a personalized greeting fragment 606, a shopping cart fragment 608, and an ad service fragment 610. The fragments depicted in **Figure 6** are shown in order of increasing rate of change for the underlying content in the fragment with product gif URL 602 having the slowest rate of change and ad service fragment 610 having the fastest rate of change. Product gif URL fragment 602 contains a hypertext reference (HREF) that specifies the URL of the file for an image that shows what the product looks like.

Product detail fragment 604 in this example may be a formatted table that includes the detailed description of the product with details, such as the product order number, name, options, and price. Personalized greeting fragment 606 is a fragment including a greeting that is directed towards the user, such as, for example, "Hello, John! Welcome to AcmeCorp.". Shopping cart fragment 608

Docket No. RSW920010011US1

in this example is a formatted shopping cart, including the order number, name, quantity and price of the products that have been chosen for possible purchase.

Ad service fragment 610 includes a HREF for an image that displays an advertisement. The advertisement HREF is different each time a page is sent to a shopper. This makes page 600 as a whole too volatile to cache. However, fragment granularity allows the other portions of page 600 to be cached.

10 The HREF to the product image in product gif URL fragment 602 and the detailed product description in product detail table fragment 604 are excellent candidates for fragments to be cached because the underlying data of a particular product changes
15 infrequently. However, the underlying data of some product changes far too frequently for static publishing.

The personalized greeting in personalized greeting fragment 606 has the lifetime of a user session, but only for a particular shopper. It may be used several times
20 within a fairly short time interval. Thus, personalized greeting fragment 606 is a good candidate for dynamic caching. Shopping cart fragment 608 changes multiple times within a user session (every time something is added or the quantity changes), so it is not as good a
25 candidate for dynamic caching as the personalized greeting. If, however, shopping cart fragment 608 is included on every page returned to the shopper, then shopping cart fragment 608 is typically returned several

Docket No. RSW920010011US1

times between changes, so there is a reasonable case for caching it. The advertisement HREF in ad service fragment 610 is a poor candidate for caching because the hit ratio would be zero and caching has its own overhead (i.e., storing it in the cache and invalidating it). Of course, each child fragment may contain additional fragments.

Figure 7 is a high level flow chart which depicts the creation of a servlet and a servlet element which uniquely identifies a cacheable servlet in accordance with the present invention. The process starts as illustrated by block 700 and thereafter passes to block 702 which depicts a programmer writing a servlet and the storage of the servlet on the application server disk. The servlet is identified by its URI. Next, block 704 illustrates the specification of cache options for each servlet. These cache options may include everything a dynamic cache will need both to correctly store dynamic data in the cache and to generate a correct cache identifier (cache ID) from the information supplied to that servlet. Thereafter, block 706 depicts the creation of a servlet element for each cacheable servlet. Each servlet element includes the servlet's URI and an indication of the cache options specified for that servlet. Block 708, then, illustrates generating a servletcache.xml file for a server which includes the servlet element for each servlet stored on this server. The process then terminates as depicted by block 710.

Docket No. RSW920010011US1

Figure 8 is a high level flow chart which illustrates processing a sample external request for a servlet in accordance with the present invention. The process starts as depicted by block **800** and thereafter passes to block **801** which depicts initializing an application server and parsing a cache policy file, `servletcache.xml`. The `servletcache.xml` file is stored on the application server disk and includes a servlet element for each cacheable servlet stored on the server. Next, block **802** which illustrates the receipt of an external request for a servlet from a user. The user is (in this example) identified by a user identifier (user ID). Thereafter, block **804** depicts a determination of whether or not the servlet needs to be initialized. If a determination is made that the requested servlet does need to be initialized, the process passes to block **808** which depicts initializing the requested servlet. The process then passes to block **810** which illustrates loading the requested servlet from the application server disk to the application server memory. Next, block **812** depicts associating cache options identified within the servlet element with the requested servlet. The process then passes to block **814**.

Referring again to block **804**, if a determination is made that the requested servlet does not need to be initialized, the process passes to block **814**. Block **814** depicts generating a cache identifier (cache ID) for the requested servlet. The cache identifier is generated

Docket No. RSW920010011US1

using the cache options for the particular servlet and the user ID. Next, block 816 illustrates a determination of whether or not any existing cache entries have this particular cache ID. The first time a user retrieves a page including a particular servlet, no cache entry will exist having the particular cache ID. If a determination is made that no cache entry exists with the particular cache ID, the process passes to block 818 which depicts retrieving the requested servlet. Next, block 820 illustrates providing the user ID and other information to the servlet. Thereafter, block 822 depicts the servlet executing and generating an HTML output. The servlet may also generate internal requests. Block 824 illustrates saving the HTML output using the cache ID as its identifier. This is then a cache entry which is identified by the particular cache ID. The process then passes to block 826 which depicts returning the HTML output to the user by displaying this cache entry.

Referring again to block 816, if a determination is made that an existing cache entry is associated with this cache ID, the process passes to block 828 which illustrates retrieving the HTML output saved as a cache entry identified by this particular cache ID. The process then passes to block 826.

Figure 9 is a high level flow chart which illustrates processing an internal request for a servlet in accordance with the present invention. The process starts as depicted by block 900 and thereafter passes to

Docket No. RSW920010011US1

block 901 which depicts initializing an application server and parsing the cache policy file. Next, block 902 illustrates the receipt of an internal request for a servlet. The input variables are defined. Next, block 904 depicts a determination of whether or not the requested servlet needs to be initialized. If a determination is made that the requested servlet does need to be initialized, the process passes to block 908 which depicts initializing the requested servlet. The process then passes to block 910 which illustrates loading the requested servlet from the application server disk to the application server memory. Next, block 912 depicts associating cache options identified within the servlet element with the requested servlet. The process then passes to block 914.

Referring again to block 904, if a determination is made that the requested servlet does not need to be initialized, the process passes to block 914. Block 914 depicts generating a cache identifier (cache ID) for the requested servlet. The cache identifier is generated using the cache options for the particular servlet and the input variables. Next, block 916 illustrates a determination of whether or not any existing cache entries have this particular cache ID. If a determination is made that no cache entry exists with the particular cache ID, the process passes to block 918 which depicts retrieving the requested servlet. Next, block 920 illustrates providing the user ID and other

Docket No. RSW920010011US1

information to the servlet. Thereafter, block 922 depicts the servlet executing and generating an HTML output. Block 924 illustrates saving the HTML output using the cache ID as its identifier. This is then a
5 cache entry which is identified by the particular cache ID. The process then passes to block 926 which depicts returning the HTML output to the requesting servlet. Referring again to block 916, if a determination is made that an existing cache entry is associated with this
10 cache ID, the process passes to block 928 which illustrates retrieving the HTML output saved as a cache entry identified by this particular cache ID. The process then passes to block 926.

It is important to note that while the present
15 invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions
20 and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such a floppy disc, a hard
25 disk drive, a RAM, CD-ROMs, and transmission-type media such as digital and analog communications links.

The description of the present invention has been presented for purposes of illustration and description,

Docket No. RSW920010011US1

and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in
5 order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
22